Domain-Adversarial Training of Neural Networks

Yaroslav Ganin GANIN@SKOLTECH.RU
Evgeniya Ustinova EVGENIYA.USTINOVA@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)
Skolkovo, Moscow Region, Russia

Hana Ajakan HANA.AJAKAN.1@ULAVAL.CA
Pascal Germain PASCAL.GERMAIN@IFT.ULAVAL.CA
Département d'informatique et de génie logiciel, Université Laval
Québec, Canada, G1V 0A6

Hugo Larochelle HUGO.LAROCHELLE@USHERBROOKE.CA
Département d'informatique, Université de Sherbrooke
Québec, Canada, J1K 2R1

François Laviolette FRANCOIS.LAVIOLETTE@IFT.ULAVAL.CA
Mario Marchand MARIO.MARCHAND@IFT.ULAVAL.CA
Département d'informatique et de génie logiciel, Université Laval
Québec, Canada, G1V 0A6

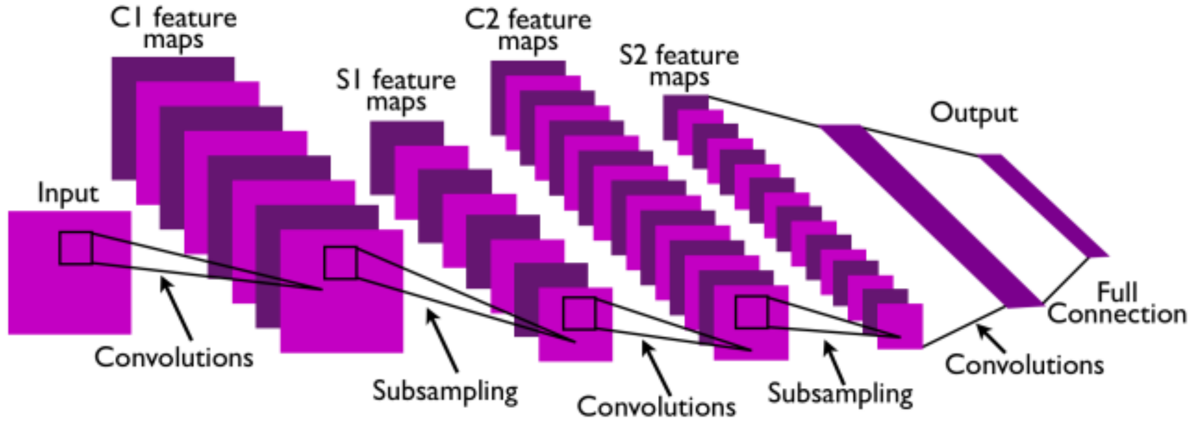
Victor Lempitsky LEMPITSKY@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)
Skolkovo, Moscow Region, Russia

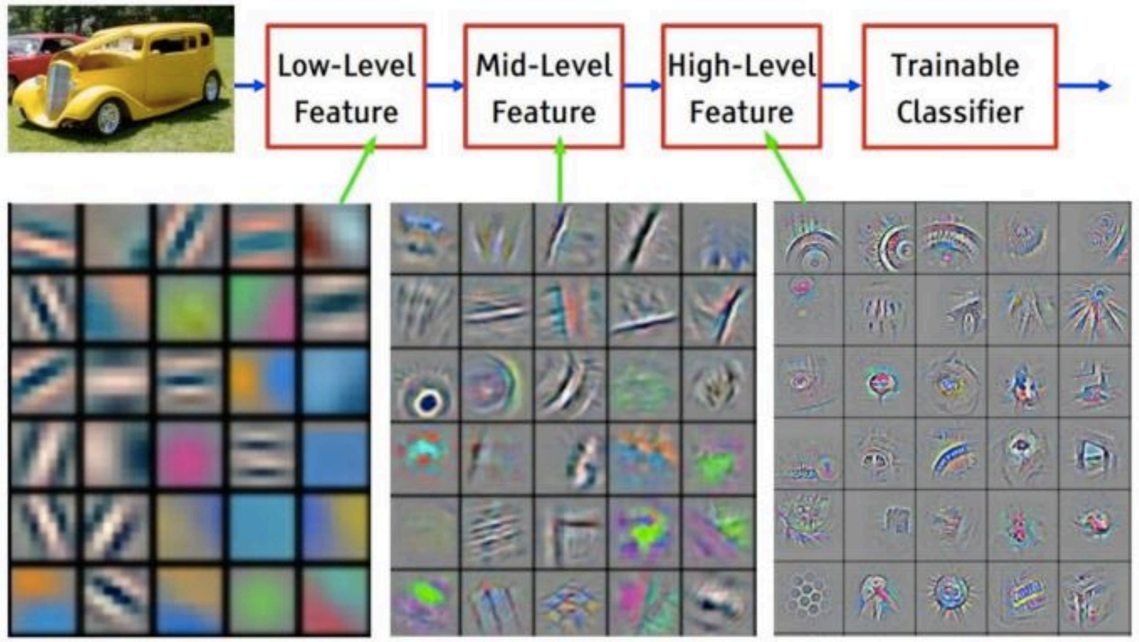
Journal of Machine Learning Research 17 (2016) 1-35 Submitted 5/15; Published 4/16

Contents

Basic concepts

- Transfer learning





Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNN에서의 학습

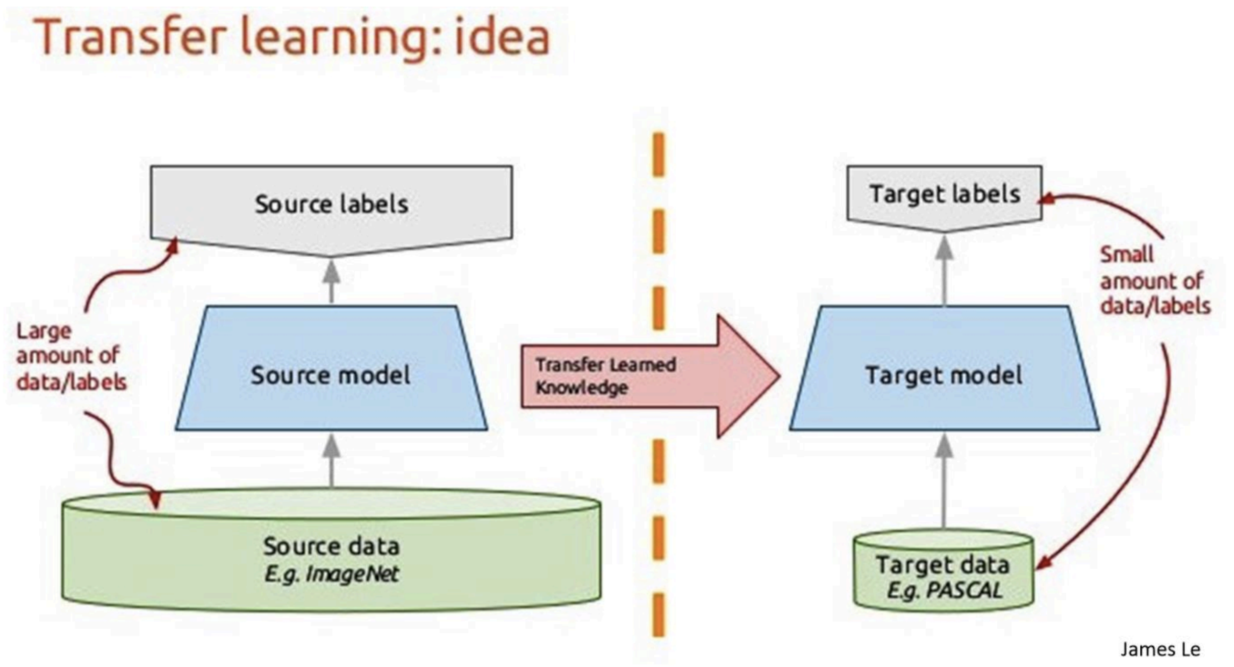
- CNN을 통하여 Feature (Weight) 를 학습한다.

- CNN layer의 앞쪽은 Low-level feature 를 학습
  Low-level feature : Edge, Corner, Color, ...
  Object의 요소가 될 수 있는 Feature

- 뒤쪽은 High-level feature 를 학습
  High-level feature : Object, Object의 일부분
  구체적인 Object에 관하여

Basic concepts

- Transfer learning



<Pretraining>　　<Fine-tuning>

Transfer learning

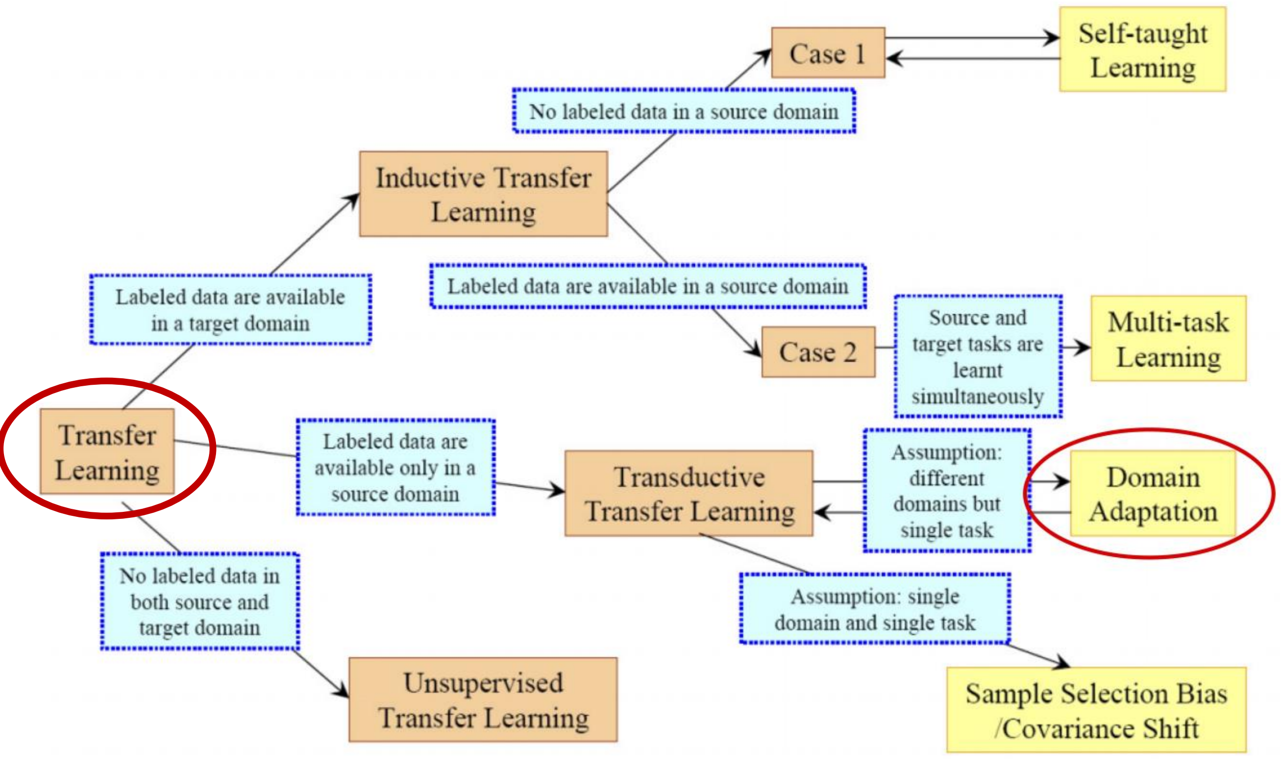
- 이미 학습된 Network의 Knowledge를 다른 작업에 사용 하는 행위

- 많은 양의 데이터를 가진 Source domain을 학습하는 단계인 'Pretraining'

- 상대적으로 적은 data가진 Target domain을 학습하는 단계인 'Fine tuning'

Basic concepts

- Transfer learning


Transfer Learning
Labeled data are available in a target domain
Inductive Transfer Learning
No labeled data in a source domain
Case 1
Self-taught Learning
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
Assumption: single domain and single task
Sample Selection Bias /Covariance Shift
No labeled data in both source and target domain
Unsupervised Transfer Learning

Transfer learning

Transfer learning은 특정 Task에 학습된 Network의 Knowledge를 다른 작업에 사용 하는 행위를 말한다.

Domain Adaptation

- Transfer learning의 한 종류
- No Target data's label
- Source <-> Target 은 다른 domain에 존재
- Single task, Source와 Target은 같은 task

Basic concepts

- Domain adaptation

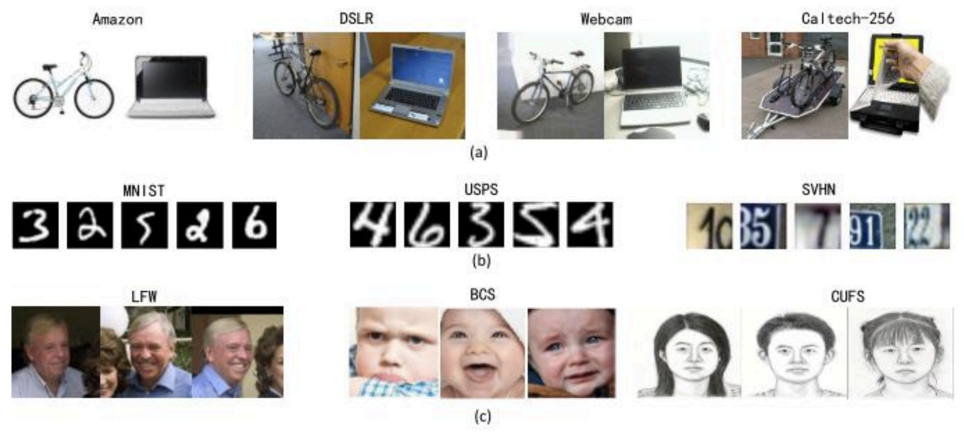
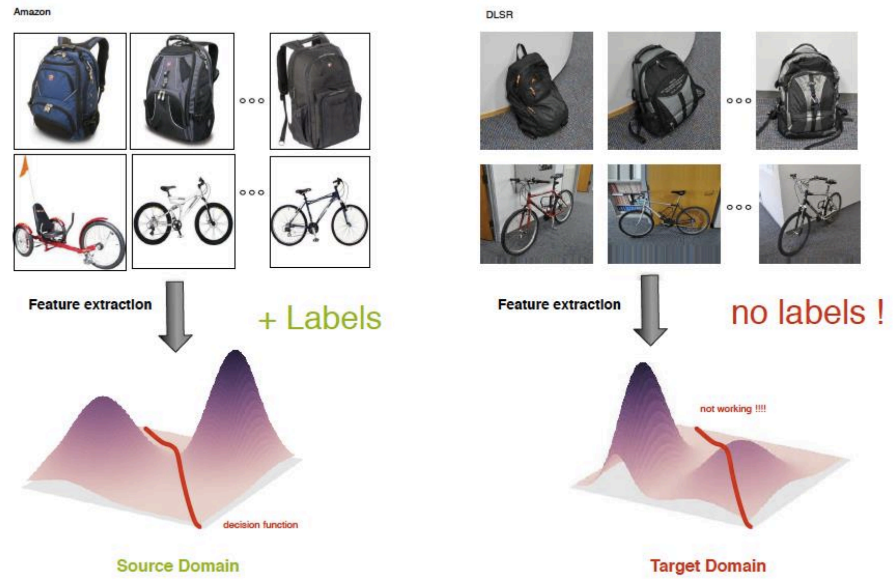


Fig. 1. (a) Some object images from the "Bike" and "Laptop" categories in Amazon, DSLR, Webcam, and Caltech-256 databases. (b) Some digit images from MNIST, USPS, and SVHN databases. (c) Some face images from LFW, BCS and CUFS databases. Realworld computer vision applications, such as face recognition, must learn to adapt to distributions specific to each domain.



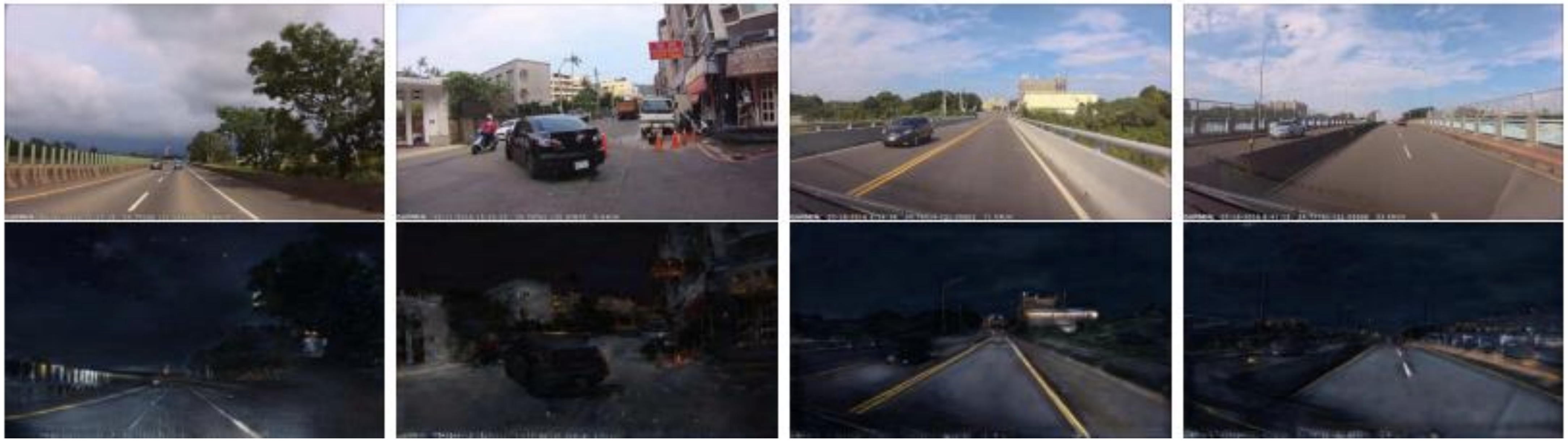
Why we need domain adaptation?

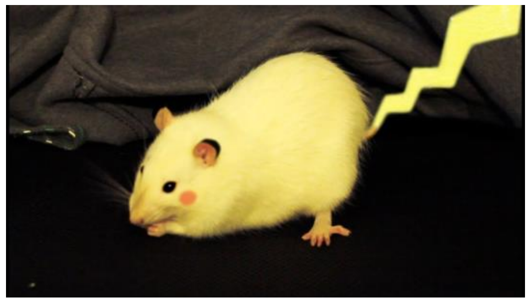
1. Domain Shift

- Fig 1. 4개의 Dataset 존재 (맨위)
  Amazon, DSLR, Webcam, Caltech, …
- 각 Dataset은 같은 label을 가지고 있다.
  Bicycle, Laptop, …
- 이중 하나의 Dataset으로 Train 된 classifier가 존재
- Classifier는 다른 Dataset에서도 효과적일까?
- 대부분의 경우 Source에서 만큼의 성능 X
- label이 같더라도 Data의 Distribution이 다르기 때문

Basic concepts

- Domain adaptation





Basic concepts

- Domain adaptation

(a) (b) (c)

Why we need domain adaptation?

2. hard to labeling

- Labeling 은 굉장히 귀찮고 시간이 많이 드는 작업

- 전문가의 도움을 받아야 하는 domain들도 다수 존재

- Label된 data가 많으면 좋겠지만
  Real-world에서는 그런 경우가 매우 드물다.
  Ex. 병원을 처음 오는 환자, Self-driving car 초행길

Basic concepts

- Domain adaptation

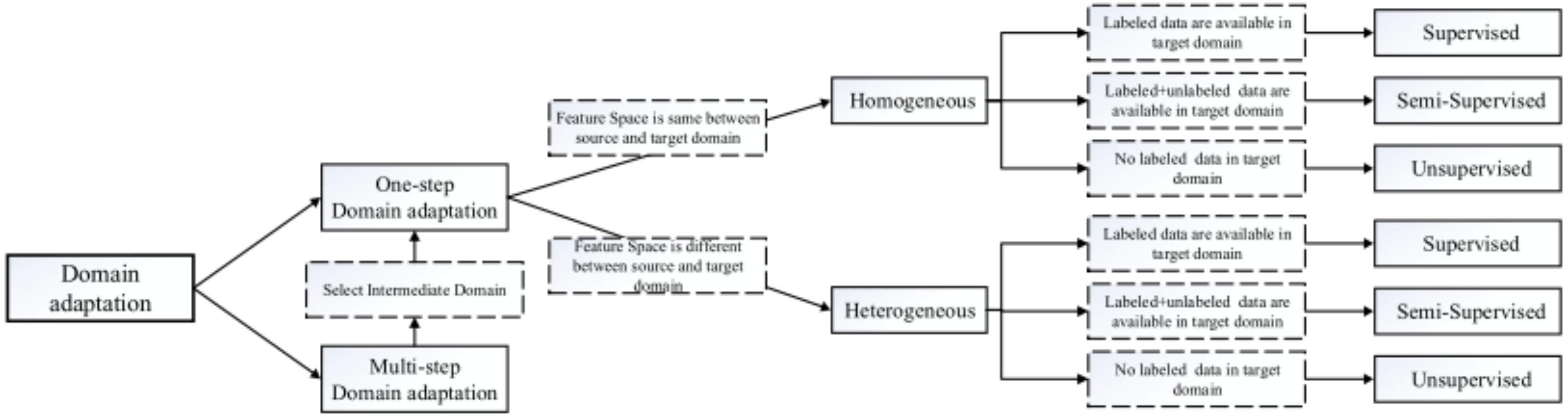

Domain adaptation
One-step Domain adaptation
Select Intermediate Domain
Multi-step Domain adaptation
Feature Space is same between source and target domain
Feature Space is different between source and target domain
Homogeneous
Heterogeneous
Labeled data are available in target domain
Labeled+unlabeled data are available in target domain
No labeled data in target domain
Supervised
Semi-Supervised
Unsupervised

Fig. 2. An overview of different settings of domain adaptation

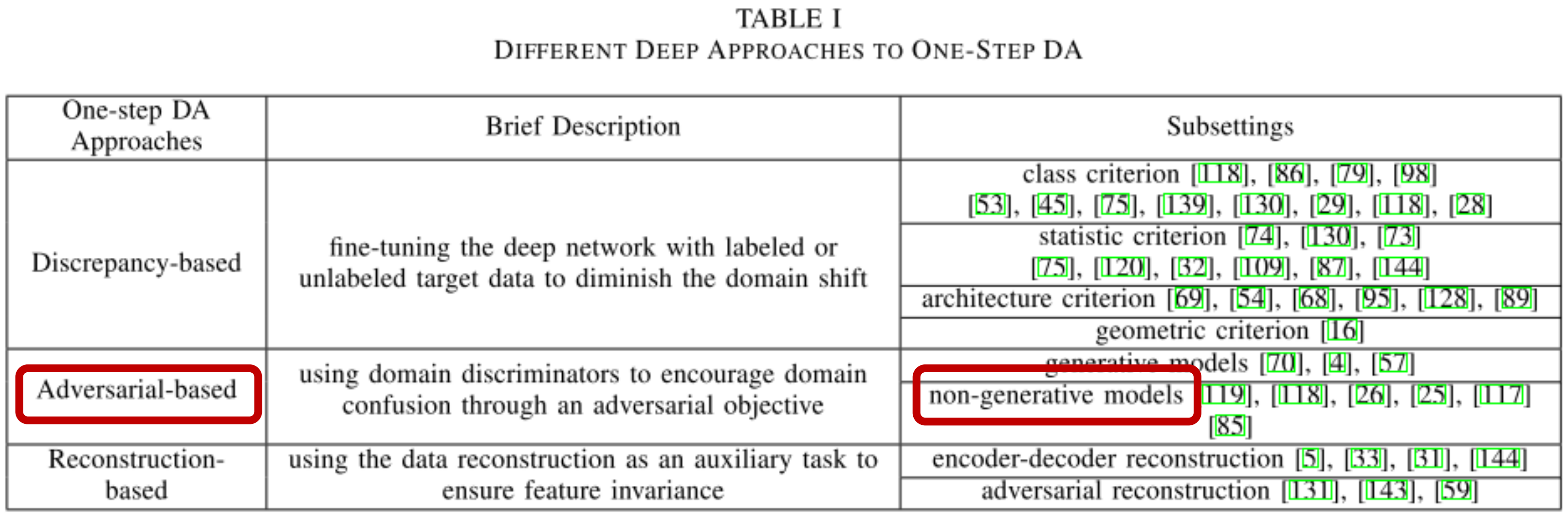
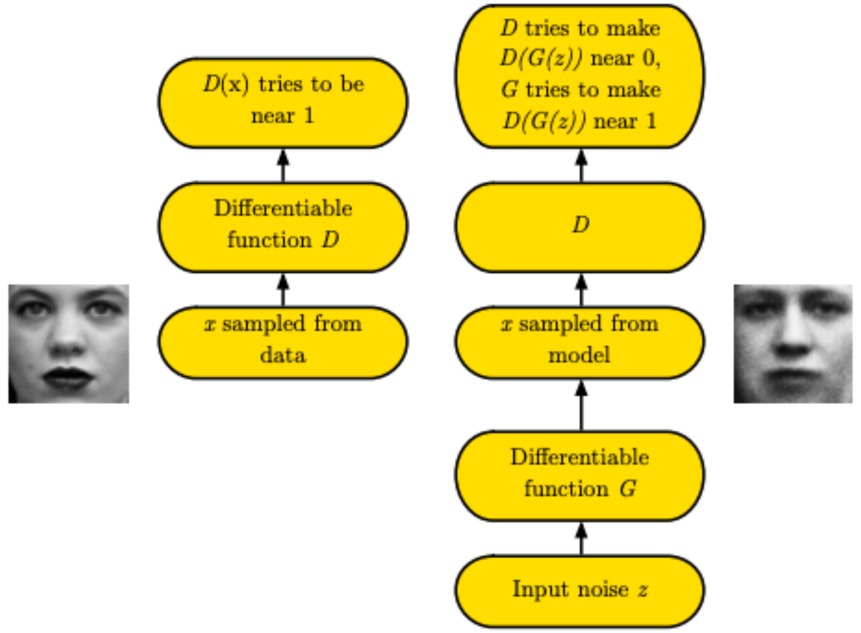


TABLE I
DIFFERENT DEEP APPROACHES TO ONE-STEP DA

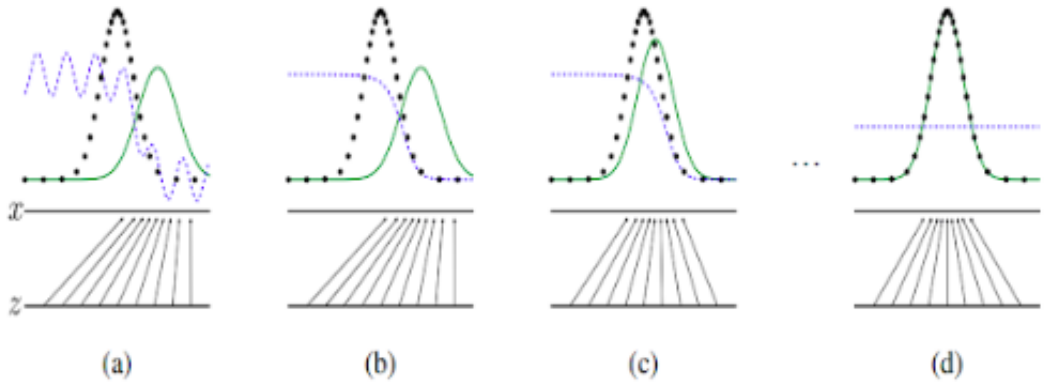
One-step DA Approaches	Brief Description	Subsettings
Discrepancy-based	fine-tuning the deep network with labeled or unlabeled target data to diminish the domain shift	class criterion [118], [86], [79], [98] [53], [45], [75], [139], [130], [29], [118], [28]
		statistic criterion [74], [130], [73] [75], [120], [32], [109], [87], [144]
		architecture criterion [69], [54], [68], [95], [128], [89]
		geometric criterion [16]
Adversarial-based	using domain discriminators to encourage domain confusion through an adversarial objective	generative models [70], [4], [57]
		non-generative models [119], [118], [26], [25], [117] [85]
Reconstruction-based	using the data reconstruction as an auxiliary task to ensure feature invariance	encoder-decoder reconstruction [5], [33], [31], [144]
		adversarial reconstruction [131], [143], [59]

Basic concepts

- GAN



D(x) tries to be near 1
Differentiable function D
x sampled from data
D tries to make D(G(z)) near 0, G tries to make D(G(z)) near 1
D
x sampled from model
Differentiable function G
Input noise z



- Generative model 'G' 와 Discriminator model 'D' 존재
- 'G'는 data의 distribution을 학습하여 유사하게 생성
- 'D'는 'G'가 생성한 distribution과 실제 data의 distribution을 구별해 내는 역할
- 따라서 'G'는 'D'가 구별해내지 못하도록 distribution 학습
  'D'는 잘 구분하도록 학습
- 두번째 그림의 검은 점선이 Data distribution
  파란 점선이 'D', 녹색 실선이 'G' 이다.
- 파란 점선은 녹색 실선과 검은 점선을 구별해 내려 한다
- 하지만 'G'는 'D'가 검은 점선과 녹색 실선을 구별하기 힘들게 학습을 한다
- 결국에는 (d)와 같은 형태가 된다.

Basic concepts

- GAN

Kullback-Leibler Divergence

$$D_{KL}(p||q) = E[\log(p_i) - \log(q_i)] = \sum_i p_i \log \frac{p_i}{q_i}$$

Non - Symmetric

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

Jensen-Shannon Divergence

$$\mathrm{JSD}(p,q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$$

데이터 분포 간의 유사도

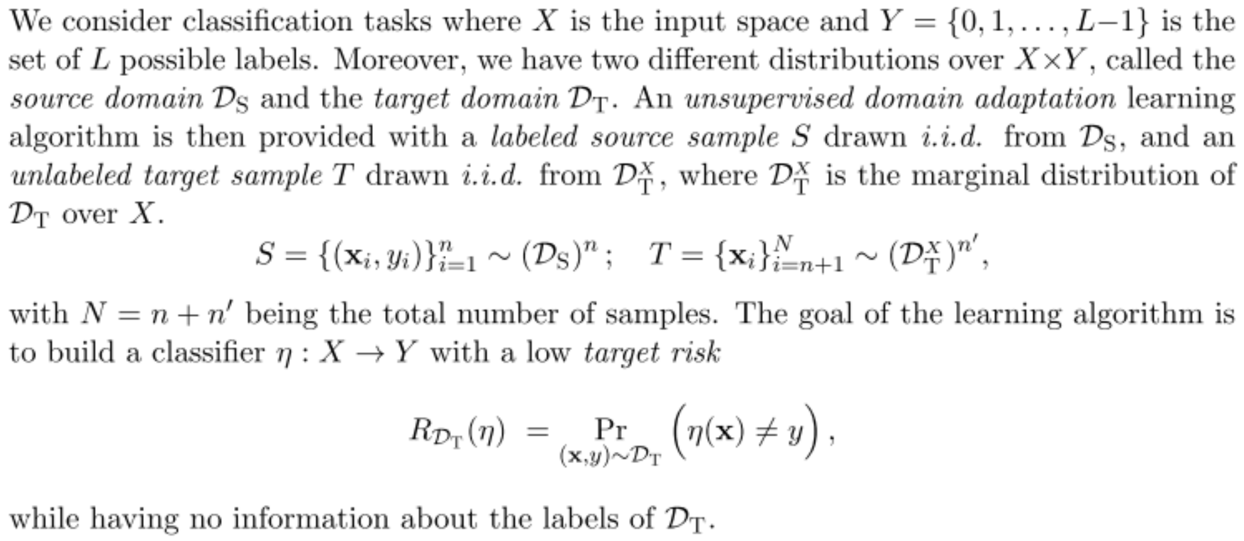
Divergence는 두 데이터의 확률 분포의 차이를 나타낸다.

Kullback-Leibler Divergence (KL Divergence)
두 모델(두 데이터)의 확률 분포가 얼마나 유사한지 알 수 있는 척도
값이 적을수록 두 확률 분포가 유사하다.

- **Jensen-Shannon Divergence**
KL Divergence를 보완한 Divergence
KL Divergence 는 Symmetric 하지 않은데, 이를 보완하여 Symmetric한 꼴로 만듦.
두 확률 분포 사이의 Distance 역할을 한다.

Basic concepts

- Basic Notation



We consider classification tasks where X is the input space and $Y = \{0, 1, \ldots, L-1\}$ is the set of L possible labels. Moreover, we have two different distributions over $X \times Y$, called the *source domain* $\mathcal{D}_S$ and the *target domain* $\mathcal{D}_T$. An *unsupervised domain adaptation* learning algorithm is then provided with a *labeled source sample* S drawn *i.i.d.* from $\mathcal{D}_S$, and an *unlabeled target sample* T drawn *i.i.d.* from $\mathcal{D}_T^X$, where $\mathcal{D}_T^X$ is the marginal distribution of $\mathcal{D}_T$ over X.

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \sim (\mathcal{D}_S)^n ; \quad T = \{\mathbf{x}_i\}_{i=n+1}^{N} \sim (\mathcal{D}_T^X)^{n'},$$

with $N = n + n'$ being the total number of samples. The goal of the learning algorithm is to build a classifier $\eta : X \to Y$ with a low *target risk*

$$R_{\mathcal{D}_T}(\eta) \;=\; \Pr_{(\mathbf{x},y)\sim\mathcal{D}_T} \Big(\eta(\mathbf{x}) \neq y \Big) ,$$

while having no information about the labels of $\mathcal{D}_T$.

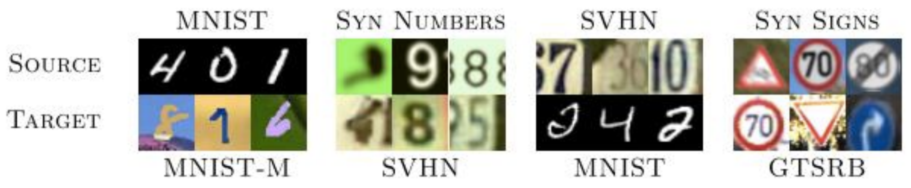
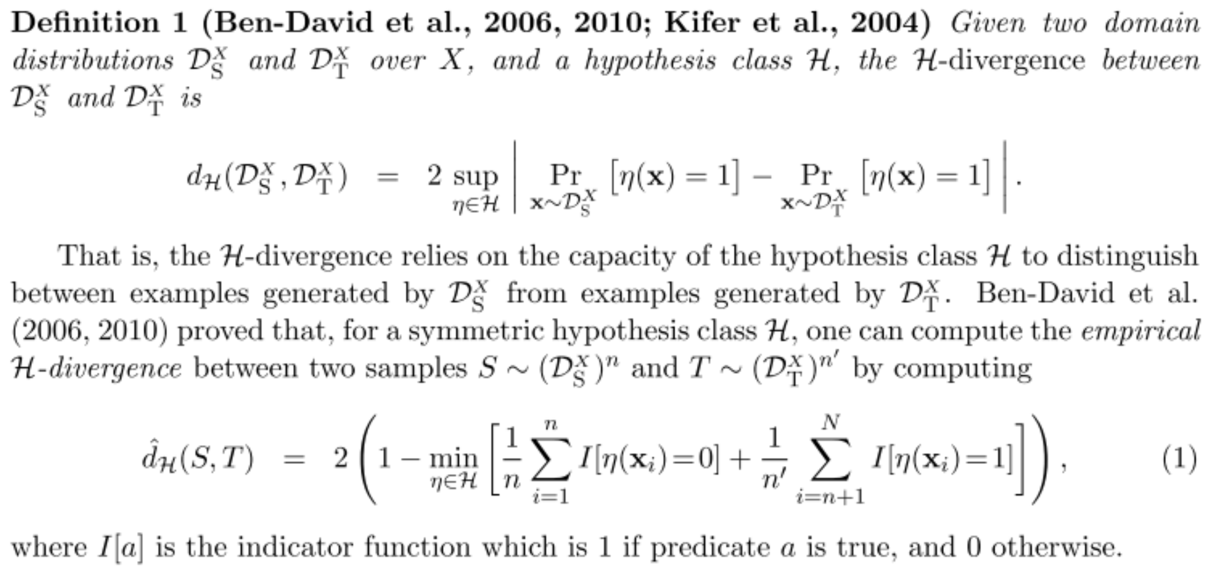


Figure 6: Examples of domain pairs used in the experiments. See Section 5.2.4 for details.

- X는 Input space (Dataset)
- Y는 Label 집합
- $D_{\{S\}}, \mathrm{D}_{\{\mathrm{T}\}}$ 는 각 Domain 을 의미한다.
- η 는 Classifier를 의미한다.
- $R_{D_T}(\eta)$ 는 Classifier의 empirical error

Basic concepts

- *H*-divergence / VC-dimension



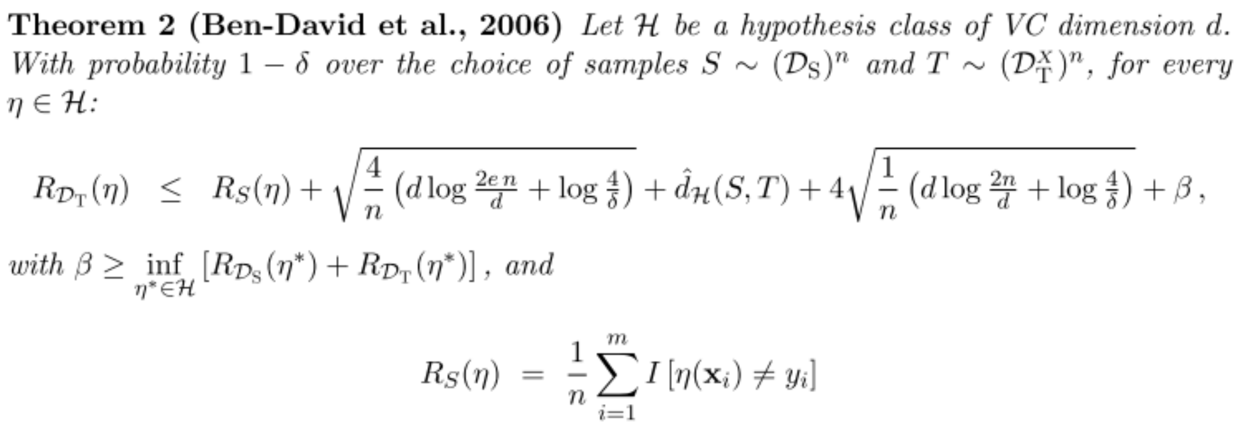
Definition 1 (Ben-David et al., 2006, 2010; Kifer et al., 2004) *Given two domain distributions $\mathcal{D}_S^X$ and $\mathcal{D}_T^X$ over X, and a hypothesis class $\mathcal{H}$, the $\mathcal{H}$-divergence between $\mathcal{D}_S^X$ and $\mathcal{D}_T^X$ is*

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) \;=\; 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x} \sim \mathcal{D}_S^X} \big[\eta(\mathbf{x}) = 1\big] - \Pr_{\mathbf{x} \sim \mathcal{D}_T^X} \big[\eta(\mathbf{x}) = 1\big] \right| .$$

That is, the $\mathcal{H}$-divergence relies on the capacity of the hypothesis class $\mathcal{H}$ to distinguish between examples generated by $\mathcal{D}_S^X$ from examples generated by $\mathcal{D}_T^X$. Ben-David et al. (2006, 2010) proved that, for a symmetric hypothesis class $\mathcal{H}$, one can compute the *empirical $\mathcal{H}$-divergence* between two samples $S \sim (\mathcal{D}_S^X)^n$ and $T \sim (\mathcal{D}_T^X)^{n'}$ by computing

$$\hat{d}_{\mathcal{H}}(S,T) \;=\; 2 \left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} I[\eta(\mathbf{x}_i) = 0] + \frac{1}{n'} \sum_{i=n+1}^{N} I[\eta(\mathbf{x}_i) = 1] \right] \right) , \qquad (1)$$

where $I[a]$ is the indicator function which is 1 if predicate a is true, and 0 otherwise.



Theorem 2 (Ben-David et al., 2006) *Let $\mathcal{H}$ be a hypothesis class of VC dimension d. With probability $1-\delta$ over the choice of samples $S \sim (\mathcal{D}_S)^n$ and $T \sim (\mathcal{D}_T^X)^n$, for every $\eta \in \mathcal{H}$:*

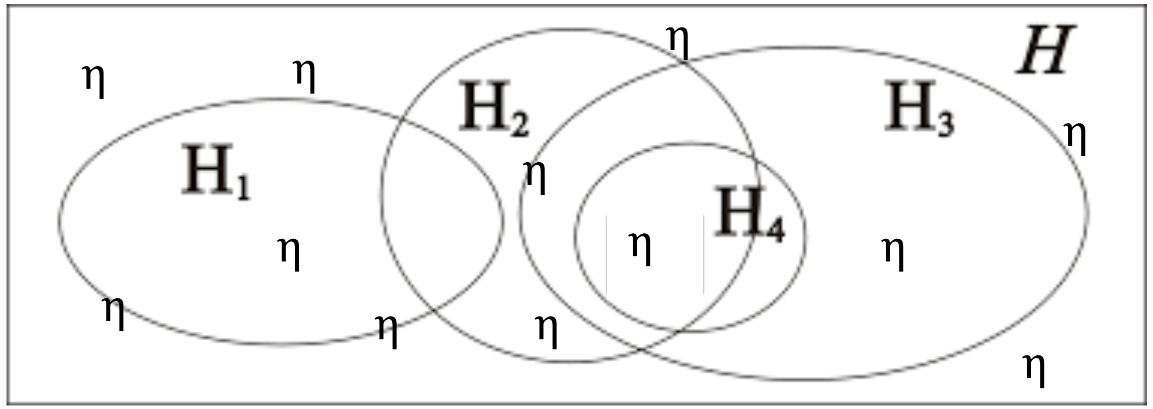
$$R_{\mathcal{D}_T}(\eta) \;\le\; R_S(\eta) + \sqrt{\frac{4}{n} \left(d \log \tfrac{2e\,n}{d} + \log \tfrac{4}{\delta} \right)} + \hat{d}_{\mathcal{H}}(S,T) + 4\sqrt{\frac{1}{n} \left(d \log \tfrac{2n}{d} + \log \tfrac{4}{\delta} \right)} + \beta\,,$$

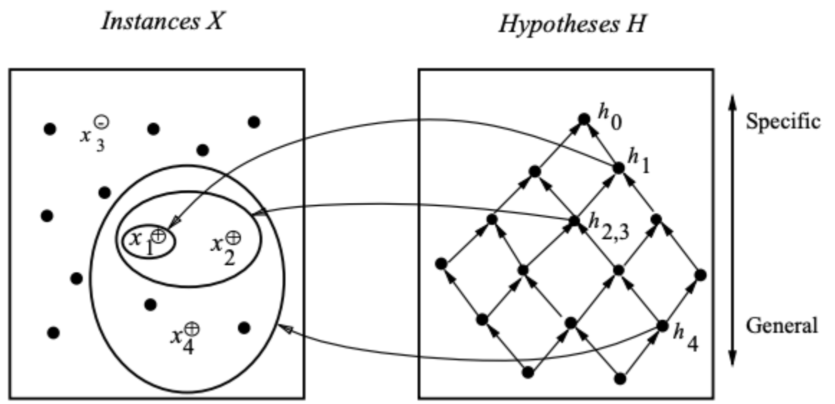
with $\beta \ge \inf_{\eta^* \in \mathcal{H}} \left[R_{\mathcal{D}_S}(\eta^*) + R_{\mathcal{D}_T}(\eta^*) \right]$, and

$$R_S(\eta) \;=\; \frac{1}{n} \sum_{i=1}^{m} I\left[\eta(\mathbf{x}_i) \neq y_i\right]$$

Basic concepts

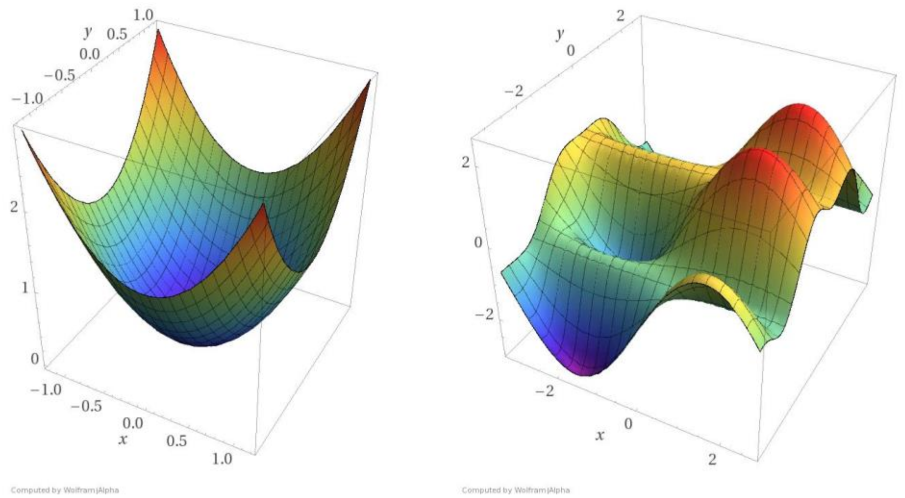
- *H*-divergence





x_1 = <Sunny Warm Normal Strong Warm Same>, +
x_2 = <Sunny Warm High Strong Warm Same>, +
x_3 = <Rainy Cold High Strong Warm Change>, -
x_4 = <Sunny Warm High Strong Cool Change>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
h_1 = <Sunny Warm Normal Strong Warm Same>
h_2 = <Sunny Warm ? Strong Warm Same>
h_3 = <Sunny Warm ? Strong Warm Same>
h_4 = <Sunny Warm ? Strong ? ? >



- Hypothesis space는 Classifier가 존재하는 공간

- Model complexity라 생각하면 됨
  ex) Linear regression -> limited to linear functions as its hypothesis space

- 일종의 Hyperparameter

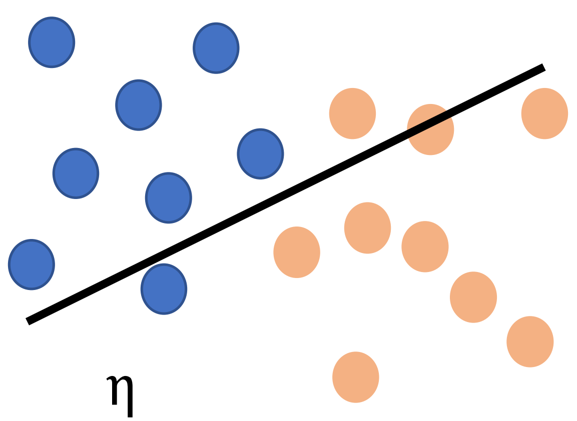
Basic concepts

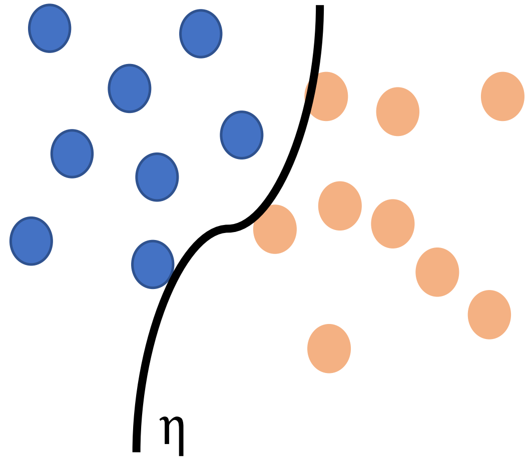
- *H*-divergence

$$d_{\mathcal{H}}(\mathcal{D}_S^X, \mathcal{D}_T^X) = 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x} \sim \mathcal{D}_S^X} [\eta(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \sim \mathcal{D}_T^X} [\eta(\mathbf{x}) = 1] \right| .$$

$$\hat{d}_{\mathcal{H}}(S, T) = 2 \left(1 - \min_{\eta \in \mathcal{H}} \left[\frac{1}{n} \sum_{i=1}^{n} I[\eta(\mathbf{x}_i) = 0] + \frac{1}{n'} \sum_{i=n+1}^{N} I[\eta(\mathbf{x}_i) = 1] \right] \right) ,$$

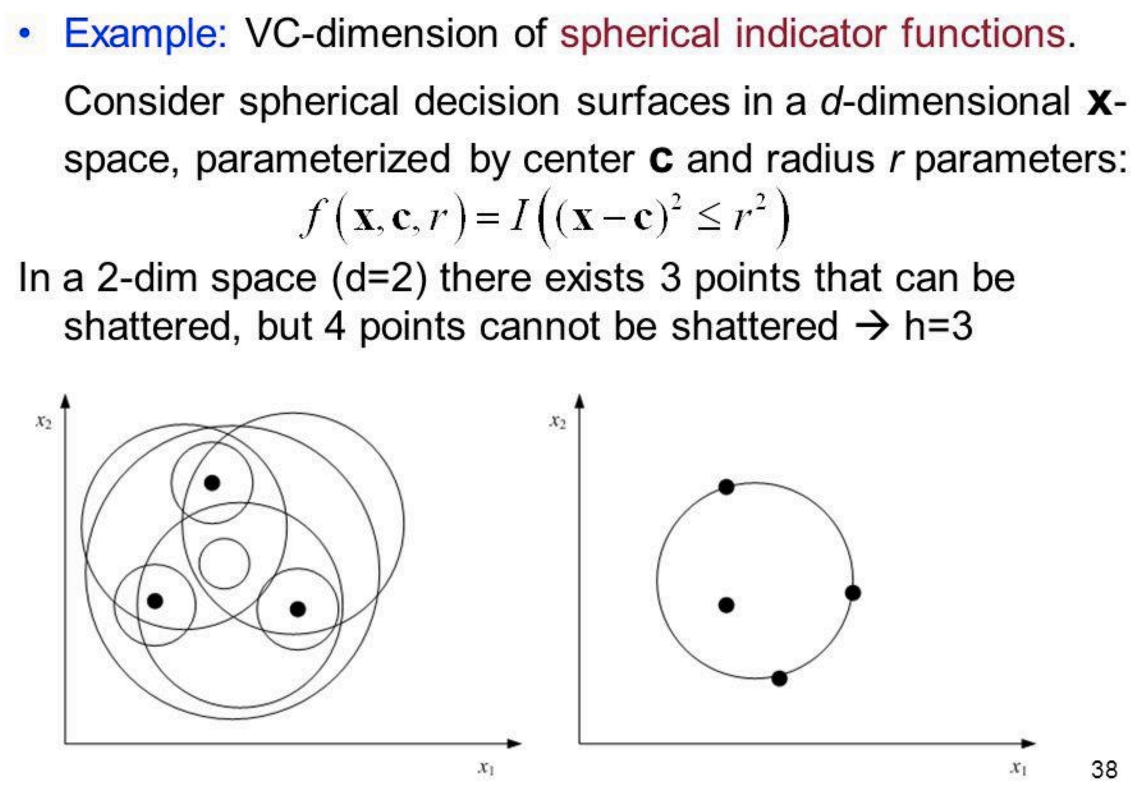
- Divergence -> data distribution을 parameter로
- H space(η 가 존재하는) 에 영향을 받는다.
- H의 값이 커지면 두 distribution을 잘 구분
- 따라서 *H*-divergence 값도 커진다.





Basic concepts

- VC Dimension



- Example: VC-dimension of spherical indicator functions.

  Consider spherical decision surfaces in a *d*-dimensional **x**-space, parameterized by center **c** and radius *r* parameters:

$$f(\mathbf{x}, \mathbf{c}, r) = I\left((\mathbf{x} - \mathbf{c})^2 \le r^2\right)$$

In a 2-dim space (d=2) there exists 3 points that can be shattered, but 4 points cannot be shattered → h=3

38

- Classification algorithm의 capacity 측정

- VC(H) 란 현재 H-space에서 label에 상관없이 정확하게 분류 할 수 있는 가장 큰 X (Data)

- 왼쪽 그림의 경우 H = 3 이다.

http://sanghyukchun.github.io/66/ 참조

Basic concepts

Theorem 2 (Ben-David et al., 2006) *Let $\mathcal{H}$ be a hypothesis class of VC dimension d. With probability $1-\delta$ over the choice of samples $S \sim (\mathcal{D}_{\mathrm{S}})^n$ and $T \sim (\mathcal{D}_{\mathrm{T}}^{\mathrm{X}})^n$, for every $\eta \in \mathcal{H}$:*

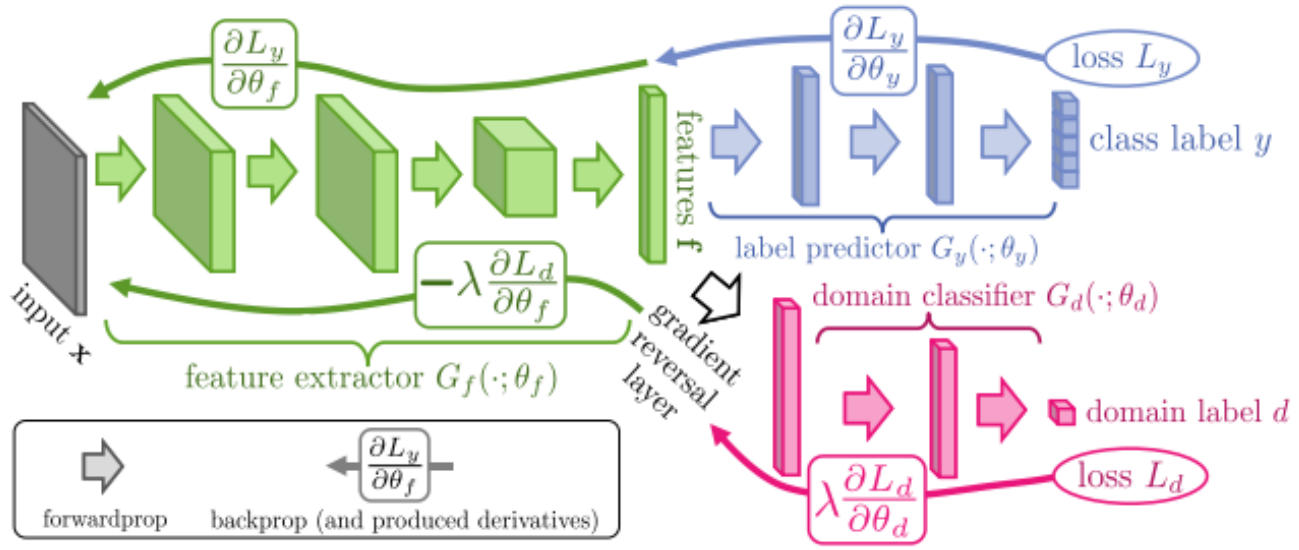
Minimize

$$R_{\mathcal{D}_{\mathrm{T}}}(\eta) \;\le\; \boxed{R_S(\eta) + \sqrt{\frac{4}{n}\left(d \log \tfrac{2e\,n}{d} + \log \tfrac{4}{\delta}\right)}} + \boxed{\hat{d}_{\mathcal{H}}(S,T) + 4\sqrt{\frac{1}{n}\left(d \log \tfrac{2n}{d} + \log \tfrac{4}{\delta}\right)}} + \boxed{\beta\,,}$$

with $\beta \ge \inf_{\eta^* \in \mathcal{H}} \left[R_{\mathcal{D}_{\mathrm{S}}}(\eta^*) + R_{\mathcal{D}_{\mathrm{T}}}(\eta^*)\right]$, *and*

$$R_S(\eta) \;=\; \frac{1}{n}\sum_{i=1}^{m} I\left[\eta(\mathbf{x}_i) \neq y_i\right]$$

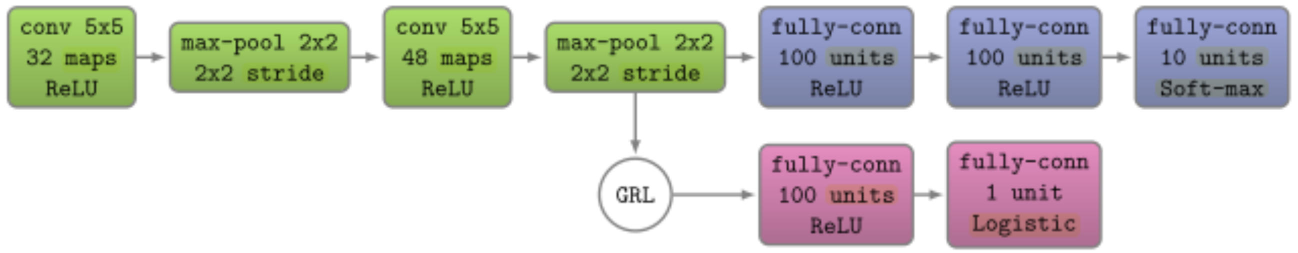
Method



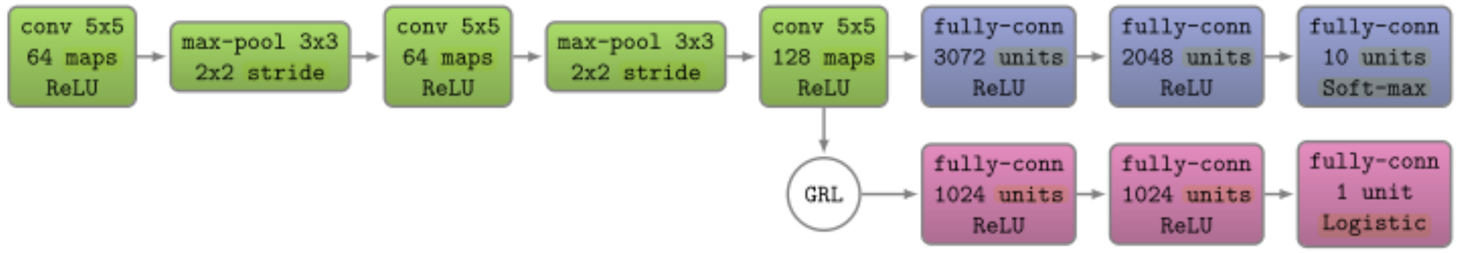
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 1: The **proposed architecture** includes a deep *feature extractor* (green) and a deep *label predictor* (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a *domain classifier* (red) connected to the feature extractor via a *gradient reversal layer* that multiplies the gradient by a certain negative constant during the backpropagation-based training. Otherwise, the training proceeds standardly and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

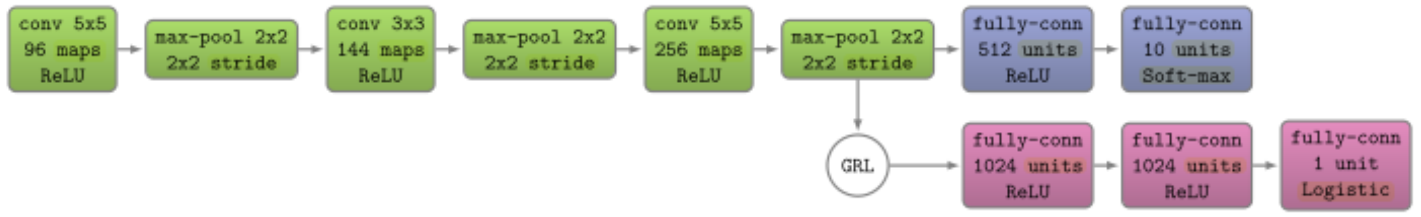
Method



(a) MNIST architecture; inspired by the classical LeNet-5 (LeCun et al., 1998).



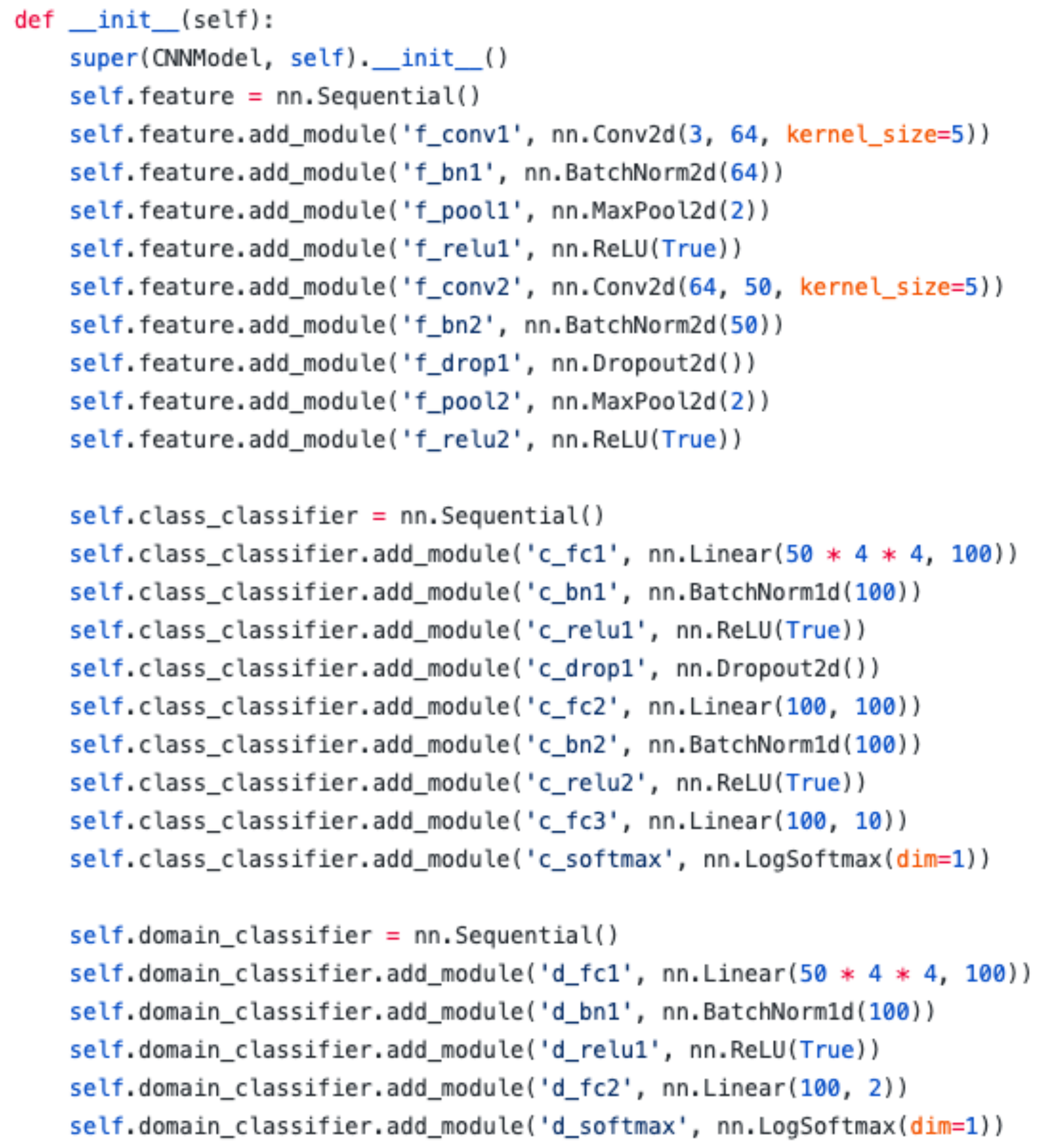
(b) SVHN architecture; adopted from Srivastava et al. (2014).



(c) GTSRB architecture; we used the single-CNN baseline from Cireşan et al. (2012) as our starting point.

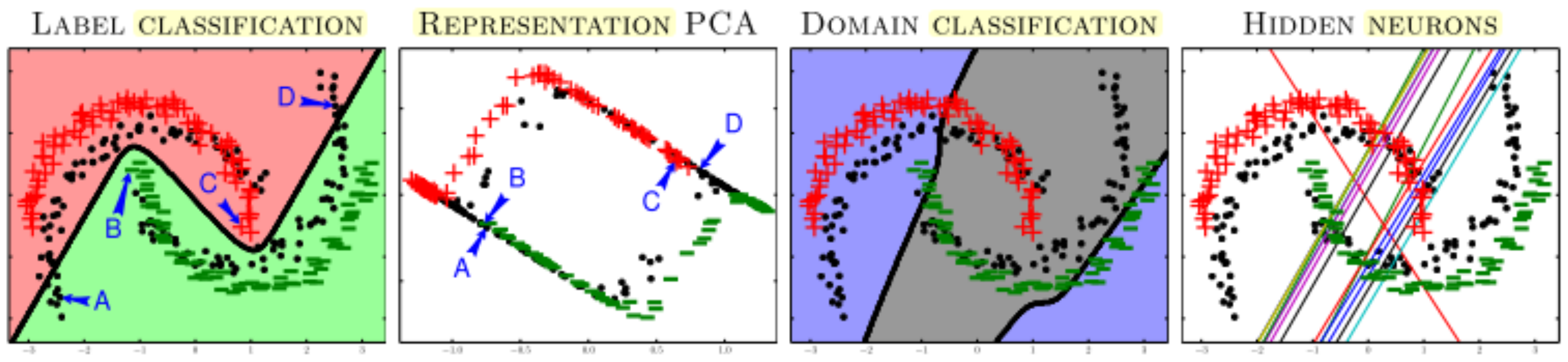
Figure 4: CNN architectures used in the experiments. Boxes correspond to transformations applied to the data. Color-coding is the same as in Figure 1.

Method

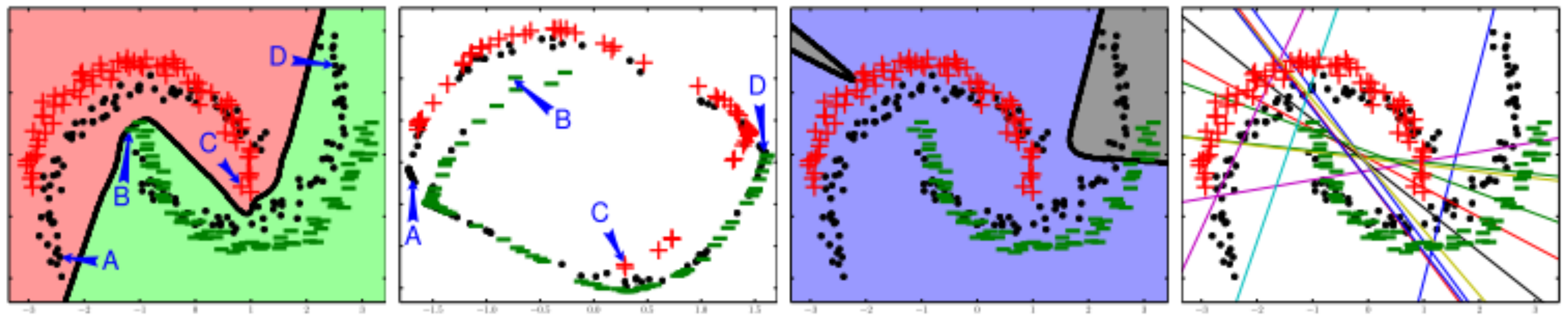


```python
def __init__(self):
    super(CNNModel, self).__init__()
    self.feature = nn.Sequential()
    self.feature.add_module('f_conv1', nn.Conv2d(3, 64, kernel_size=5))
    self.feature.add_module('f_bn1', nn.BatchNorm2d(64))
    self.feature.add_module('f_pool1', nn.MaxPool2d(2))
    self.feature.add_module('f_relu1', nn.ReLU(True))
    self.feature.add_module('f_conv2', nn.Conv2d(64, 50, kernel_size=5))
    self.feature.add_module('f_bn2', nn.BatchNorm2d(50))
    self.feature.add_module('f_drop1', nn.Dropout2d())
    self.feature.add_module('f_pool2', nn.MaxPool2d(2))
    self.feature.add_module('f_relu2', nn.ReLU(True))

    self.class_classifier = nn.Sequential()
    self.class_classifier.add_module('c_fc1', nn.Linear(50 * 4 * 4, 100))
    self.class_classifier.add_module('c_bn1', nn.BatchNorm1d(100))
    self.class_classifier.add_module('c_relu1', nn.ReLU(True))
    self.class_classifier.add_module('c_drop1', nn.Dropout2d())
    self.class_classifier.add_module('c_fc2', nn.Linear(100, 100))
    self.class_classifier.add_module('c_bn2', nn.BatchNorm1d(100))
    self.class_classifier.add_module('c_relu2', nn.ReLU(True))
    self.class_classifier.add_module('c_fc3', nn.Linear(100, 10))
    self.class_classifier.add_module('c_softmax', nn.LogSoftmax(dim=1))

    self.domain_classifier = nn.Sequential()
    self.domain_classifier.add_module('d_fc1', nn.Linear(50 * 4 * 4, 100))
    self.domain_classifier.add_module('d_bn1', nn.BatchNorm1d(100))
    self.domain_classifier.add_module('d_relu1', nn.ReLU(True))
    self.domain_classifier.add_module('d_fc2', nn.Linear(100, 2))
    self.domain_classifier.add_module('d_softmax', nn.LogSoftmax(dim=1))
```

Result



(a) Standard NN. For the “domain classification”, we use a *non adversarial* domain regressor on the hidden neurons learned by the Standard NN. (This is equivalent to run Algorithm 1, without Lines 22 and 31)



(b) DANN (Algorithm 1)

Figure 2: The *inter-twinning moons* toy problem. Examples from the source sample are represented as a “+”(label 1) and a “−”(label 0), while examples from the unlabeled target sample are represented as black dots. See text for the figure discussion.